A Genetic Algorithm to Eliminate Disk Fragmentation

[+]Mahdi A. Salman
Computer Dept.
Collage of Science for Women

[+]Mohamed U. Mahdi
Computer Dept.
Collage of Science for Women

[+]Faez Ali Rashid
Computer Dept.
Collage of Science for Women

Abstract

Disk fragmentation is one of major problem for operating systems. It's occur when several operation of delete/write of file are made. Genetic algorithm used to determine the best way of empty area swapping with file fragments. It is make a paralleled virtual defragmentation for disk and then select the one that take a shortest time to carry out. By this method we can save a much west time of non useful movement of fragments using old methods.

Keywords: Disk fragmentation, Disk management, Genetic Algorithm, Parallel Computing.

المستخلص: استخدام الخوارزمية الجينية للتخلص من التجزئة في خزن الملفات

من المشاكل الرئيسية التي يواجهها النظام التشغيلي للحاسب هو خزن الملفات بالتجزئة. تظهر المشكلة عندما يكون هناك عمليات متعاقبة من الخزن والحذف. تم استخدام الخوارزمية الجينية لتحديد أفضل عملية تناقل ممكنة بين أجزاء الملفات والمناطق الفارغة. وتقوم الخوارزمية الجينية بالبحث المتوازي ومن ثم يتم اختيار الحل الأقل وقت للتنفيذ. بهذه الطريقة نستطيع تجنب خسارة الوقت للتحريك عديم الفائدة المستخدم بالطرق القديمة.

1. Introduction

Defragmentation (or defragging) is a process that eliminates fragmentation in file systems. It does this by physically reorganizing the contents of the disk in order to store the pieces of each file close together and in order (contiguously). It also attempts to create large regions of free space using compaction, to impede the return of fragmentation [David D. Grossman and Harrey F. Silverman,1973].

Reading and writing data on a heavily fragmented hard drive is slowed down as the time for the heads to move between fragments on the disk surface can be substantial. The disk operates at speeds millions of times slower than the CPU; thus the desire to process more efficiently encourages defragmentation. Operating system manufacturers often recommend periodic defragmentation in order to keep hard drive access as fast as possible [Chris Ruemmler and John Wilkes,1994].

Fragmentation occurs when the operating system cannot or will not allocate enough contiguous space to store a complete file as a unit, but instead puts parts of it in gaps between other files (usually those gaps exist because they formerly held a file that the operating system has subsequently deleted or because the operating system allocated excess space for the file in the first place). As advances in technology brings larger disk drives, the performance loss due to fragmentation squares with each doubling of the size of the drive. Larger files and greater numbers of files also contribute to fragmentation and consequent performance loss. Defragmentation restores a drive to its original speed. It also moves infrequently used files further from the directory area.

A defragmentation program must move files around within the free space available in order to undo fragmentation. This is a memory intensive operation and cannot be performed on a file system with no free space. The reorganization involved in defragmentation does not change logical location of the files (defined as their location within the directory structure) [Nianlong and Kelly J. flanagan. 1998].

Most of operating systems shipped with them a utility program for disk defragmentation. But they are take a long time to finish task (for example (30-40) minutes for disk defragmentation utility of Windows™ XP® for about 15GB disk size. We suggest new enhancement of disk defragment process through using GA to determine best movement of fragments.

2. Related works

Several techniques for disk rearrangement have been proposed [Paul Vongsathorn and Scott D. Carson,1990]. They reorganize the disk based on the idea that a disk's performance can be improved by clustering frequently accessed data. The methods proposed in [Sedat Akyuerek and Kenneth Salem,1997] used a technique called Organ Pipe[David D. Grossman and Harrey F. Silverman,1973]. The organ pipe heuristic places the most accessed data (hot blocks) in the center of the disk, then places the next most frequently accessed data on either side of the center; this process continues until the least-accessed data is placed at the edge of the disk. Another method proposed in [Xiao-Hong Tu, Niki C. Thornock and J.Kelly Flanagan,1997] places the hot blocks on the edge of the disk, and the least accessed data in the center of the disk.

3. Genetic algorithm

The most popular technique in evolutionary computation research has been the genetic algorithm. In the traditional genetic algorithm, the representation used is a fixed-length bit string. Each position in the string is assumed to represent a particular Feature of an individual, and the value stored in that position represents how that feature is expressed in the solution. Usually, the string is "evaluated as a collection of structural features of a solution that have little or no interactions". The analogy may be drawn directly to genes in biological organisms. Each gene represents an entity that is structurally independent of other genes [Sivanandam S.N.· Deepa S.N,2008.].

The main reproduction operator used is bit-string crossover, in which two strings are used as parents and new individuals are formed by swapping a sub-sequence between the two strings (see Fig. 1).

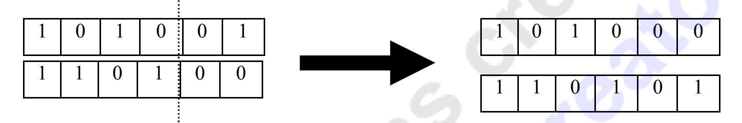


Fig (1) Crossover Operator

Another popular operator is bit-flipping mutation, in which a single bit in the string is flipped to form a new offspring string (see Fig. 2).

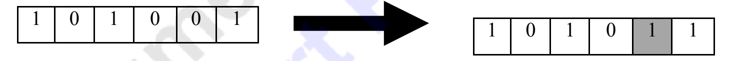


Fig (2) Mutation Operator

Varieties of other operators have also been developed, but are used less frequently (e.g., inversion, in which a subsequence in the bit string is reversed). A primary distinction that may be made between the various operators is whether or not they introduce any new information into the population. Crossover, for example, does not while mutation does. All operators are also constrained to manipulate the string in a manner consistent with the structural interpretation of genes. For example, two genes at the same location on two strings may be swapped between parents, but not combined based on their values. Traditionally, individuals are selected to be parents probabilistically based upon their fitness values, and the offspring that are created replace the parents. For example, if N parents are selected, then N offspring are generated which replace the parents in the next generation [Haupt I., Haupt S. E. II.and Randy L.,2004].

4. Suggested Algorithm

4.1 Representation:

GA handles a population of possible solutions. Each solution is represented through a chromosome, which is just an abstract representation. Coding all the possible solutions into a chromosome is the first part. Each chromosome represents the whole disk layout. Every gene represents a fragment of file k as shown in figure 3.

F1.1	F1.2	F2.1	F1.3	S	F2.3	F3.1	F3.2	F3.3	F3.4	S	F2.4	F4.3	F4.1	S	F4.2	S

Fig. 3 The encoding of Chromosome

4.2 Initialization

Each gene initialize as: randomly a fragment is taken from disk and assigned to gene. Some genes may represent space which denoted by S.

The genetic algorithm loops over an iteration process to make the population evolve. Each iteration consists of the following steps:

4.3 Evaluation

Evaluate the fitness f(x) of each chromosome x in the population as:

$$\text{Let } g(k) = \frac{\sum_{j=1}^{m-1} \left\| f_j - f_{j+1} \right\|}{m} \quad \text{..(1)}$$

Where m is number of fragments of file k, f_j is fragment j, then

$$fit(x) = \sum_{k=1}^{n} \left(g(k) - \overline{g(k)}\right)^2 \quad \text{......................................(2)}$$

Where n is number of files and $\overline{g(k)}$ is the mean

4.4 Selection

Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to get selected).

4.5 Crossover

With a crossover probability, crossover the parents to produce new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
We used a modified crossover operator by reordering genes inside the cut area of new child and discard existing gene from parents to produce only new one Child.
as illustrate in (fig. 4).

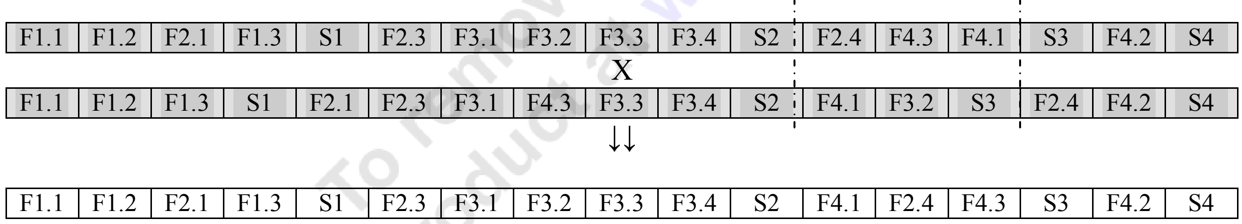


Fig.4 Crossover in Suggested Algorithm

4.6 Mutation

Mutation made by swapping two genes selected randomly of produced child. As shown in fig.5

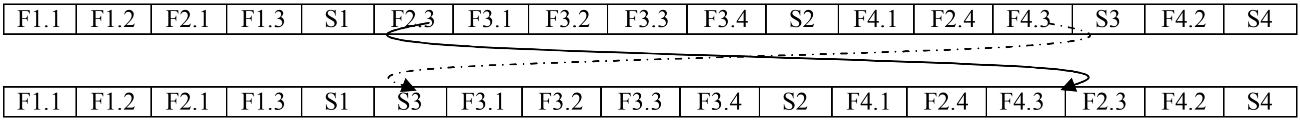


Fig.5 Mutation in Suggested Algorithm

4.7 Stopping criteria

Algorithm iterates its operator until no change in maximum fitness or after 100 generation.

4.8 Results

We randomly generate 5 simulated samples of hard disks and we got result illustrate in (table 1) for each sample after use it with suggested method.

The last column is computed as the rate of defragmeneted to fragmented files after running suggested algorithm.

Table 1 shows the results of suggested algorithms for some samples

Sample #	Number of files	Number of fragments	Rate
1	10	35	52%
2	15	50	50%
3	20	40	40%
4	25	60	39%
5	30	60	35%

5. Conclusions

In this paper, we have presented a flexible disk defragmenter system that uses genetic algorithm. It is rearrange the file fragments. We show the ability of made virtual fragmentation elimination of hard disk and determine the optimal transferring strategy of fragments.

6. References

David D. Grossman and Harrey F. Silverman “Placement of Records on a Secondary Storage Device to Minimize Access Time”, Journal of the ACM, 20(3), 429-438 (1973).

Chris Ruemmler and John Wilkes, "An introduction to disk drive modeling", IEEE Computer, 1994.

Nianlong Yin and J. Kelly Flanagan, "Reducing Application Load Time by Rearranging Disk Data", thesis, 1998.

Paul Vongsathorn and Scott D. Carson “A System for Adaptive Disk Rearrangement” Software---Practice and Experience, 20(3), pp. 225-242, 1990.

Sedat Akyuerek and Kenneth Salem “Adaptive Block Rearrangement Under UNIX” Software---Practice and Experience, 27(1), 1997.

Xiao-Hong Tu and Niki C. Thornock and J.Kelly Flanagan “A Stochastic Disk I/O Simulation Technique”, Proceedings of the 1997 Winter Simulation Conference, pp 1079 – 1086, 1997. Atlanta, Georgia, U.S.A.

Sivanandam S.N.and Deepa S.N., "Introduction to Genetic Algorithms" Springer Pub. Co., 2008.

Haupt I., Haupt S. E. II.and Randy L. "Practical genetic algorithms.", Wiley Pup. Co., 2004.